What is claimed is:

1. A method for processing a request, comprising,

receiving the request,

selecting a software thread based on the request,

upon determining that the selected software thread requires modification to process the request, updating the selected software thread to process the request, and,

processing the request using the selected software thread.

2. A method according to claim 1, wherein selecting the software thread based on the request further includes implementing a hash table to associate the request to at least one software thread.

3. A method according to claim 2, further including translating the request to a key for the hash table.

4. A method according to claim 2, wherein implementing a hash table to associate the request further includes,

translating a URL in the request to a key, and,

indexing the hash table by the key.

5. A method according to claim 1, wherein selecting a software thread based on the request further includes selecting a software thread based on the software thread that has been idle for the longest time.

6. A method according to claim 1, wherein determining that the selected software thread requires updating further includes,

verifying the software thread includes application logic to process the request.

7. A method according to claim 1, wherein determining that the selected software thread requires updating further includes, verifying the software thread includes a most recent version of application logic to process the request.

8. A method according to claim 1, wherein updating the selected software thread to process the request further includes receiving application logic to process the request.

9. A method according to claim 1, wherein updating the selected software thread to process the request further includes establishing a persistent connection between the selected software thread and at least one database.

10. A method according to claim 1, wherein updating the selected software thread to process the request further includes establishing a persistent connection between the selected software thread and at least one file system.

11. A method according to claim 1, wherein updating the selected software thread further includes byte-compiling the data for incorporation into the software thread.

12. A method according to claim 1, wherein updating the selected software thread further comprises,

retrieving data through a persistent connection,

byte-compiling the retrieved data, and,

storing the byte-compiled data in memory local to the software thread.

13. A method according to claim 1, wherein updating the selected software thread to process the request further includes establishing a persistent connection between the selected software thread and at least one session manager daemon.

14. A method according to claim 1, wherein updating the selected software thread to process the request further includes establishing a persistent connection between the selected software thread and at least one real-time data source.

15. A method according to claim 1, wherein updating the selected software thread to process the request further includes,

comparing application logic in the selected software thread to a latest version of the application logic, and,

upon determining that the latest version of the application logic is different than the application logic in the selected software thread, incorporating the latest version of the application logic into the selected software thread.

16. A method according to claim 15, wherein determining that the latest version of the application logic is different than the application logic in the selected software thread further includes, determining that the latest version of the application

software is dated more recently than the application logic in the selected software thread.

17. A method according to claim 15, wherein incorporating the latest version of the application logic into the software thread further includes, replacing the application logic in the selected software thread with the latest version of the application logic.

18. A method according to claim 1, further comprising returning the processed request to the requesting device.

19. A method according to claim 1, wherein receiving the request further includes receiving the request from a server.

20. A method according to claim 1, wherein receiving the request further includes receiving the request from a client.

21. A method according to claim 1, wherein receiving the request further includes receiving an HTTP request.

22. A method according to claim 1, wherein receiving the request further includes receiving the request from a network.

23. A method for distributing a request amongst software threads, comprising,

identifying non-processing software threads,

arranging the non-processing software threads according to use,

comparing the non-processing software threads to the request, and, based on the comparison, selecting the a software thread to process the request.

24. A method according to claim 23, wherein selecting a software thread further comprises determining at least one of a software thread that is most infrequently used while comparing to the request, and a software thread that is most infrequently used while not comparing to the request.

25. A method according to claim 23, wherein selecting a software thread further comprises determining at least one of a software thread that has a greatest time since last use while comparing to the request, or a software thread that has greatest time since last use while not comparing to the request.

26. A method according to claim 23, wherein arranging the non-processing software threads according to use includes implementing a queue.

27. A method according to claim 26, wherein implementing a queue includes providing a first-in, first-out queue.

28. A method according to claim 23, wherein arranging the non-processing software threads according to use includes implementing a doubly linked list.

29. A method according to claim 23, wherein identifying non-processing software threads further includes, polling the software threads for activity.

30. A method according to claim 23, wherein comparing the non-processing software threads to the request further includes,

generating a key from the request,

using the key as an index to a hash table to identify software threads associated with the key, and,

identifying those associated software threads that are non-processing.

31. A system for processing a request, comprising,

a first server to receive the request, and,

a plurality of software threads to process the request, the software threads being adaptable to incorporate application logic based on the request.

32. A system according to claim 31, further comprising, at least one alternate server, at least one of the alternate servers being in communication with the first server, the alternate servers capable of receiving requests from at least one of the first server and at least one alternate server.

33. A system according to claim 32, wherein the at least one alternate servers further comprise a plurality of software threads adaptable to incorporate application logic based on the request to process the received request.

34. A system according to claim 31, further comprising a decision module to associate the request to at least one of the software threads.

35. A system according to claim 34, wherein the decision module includes a hash table.

36. A system according to claim 31, wherein the request further includes a Uniform Resource Location (URL).

37. A system according to claim 31, wherein the request is received via a network.

38. A system according to claim 31, further comprising a status module to collect statistics based on the software threads.

39. A system according to claim 38, wherein the status module further includes at least one of software thread processing status, software thread processing time, software thread application code, time of last use of software thread, and software thread persistent connections.

40. A system according to claim 38, wherein the status module further comprises a queue to order the software threads.

41. A system according to claim 38, wherein the status module further comprises a doubly linked list to order the software threads.

42. A system according to claim 31, wherein the software threads further include persistent connections to at least one database.

43. A system according to claim 31, wherein the software threads further include persistent connections to at least one real-time data source.

44. A system according to claim 31, wherein the software threads further include persistent connections to at least one session manager daemon.

45. A system according to claim 31, wherein the software threads further include persistent connections to at least one file system.

46. A system according to claim 31, further comprising local memory accessible to the software threads.